(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 468 229 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **91110885.0**

(51) Int. Cl.5: **G06F 15/80**, G06F 9/44

(22) Date of filing: **01.07.91**

(30) Priority: **27.07.90 US 558755**

(43) Date of publication of application:
29.01.92 Bulletin 92/05

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IT LI LU NL SE

(71) Applicant: **HNC, INC.**
5501 Oberlin Drive
San Diego, California 92121-1718(US)

(72) Inventor: **Levey, Curt A.**
7180 Shoreline Drive
5210 San Diego, California 92122(US)

(74) Representative: **Liesegang, Roland, Dr.-Ing. et al**
FORRESTER & BOEHMERT
Widenmayerstrasse 4
W-8000 München 22(DE)

(54) A neural network with expert system functionality.

(57) A method is disclosed for performing a variety of expert system functions on any trained feedforward neural network. These functions include decision-making, explanation, computation of confidence measures, and intelligent direction of information acquisition. Additionally, the method converts the knowledge implicit in such a network into a set of explicit if-then rules.

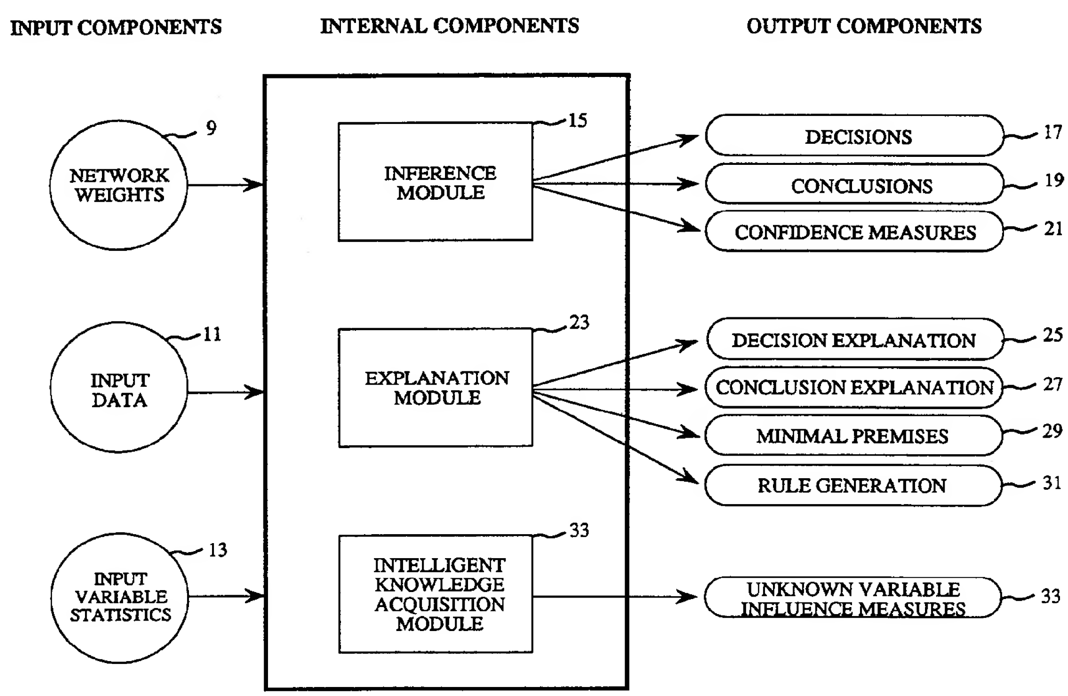

Figure 1

EP 0 468 229 A2

Field of the Invention

This invention relates to neural networks and more particularly to a class of neural networks known as feedforward networks.

Background of the Invention

Neural networks are composed of a series of interconnected neuron-like processing elements (PEs). The strengths of the connections between PEs are represented by weights. Each PE stores a value known as a state, which is either specified by input data or computed from the PE's inputs and weights, using its transfer function. Typically, the transfer function is applied to the PE's net-input, the weighted sum of its inputs. Collectively, states are used to represent information in the short term. Long-term information or learning is represented by the weights. Neural networks learn from examples by modifying their weights. Once learning or training is completed, these networks can perform a variety of computational tasks.

Much of the use of neural networks is focused on feedforward networks. These networks have an architecture consisting of a layer of input nodes, a layer of output nodes, and optionally, some number of hidden layers in between. Input data is represented using the states of the input layer. The network's response to that data is represented by the states of the output layer. The feedforward nature of these networks results from the fact that, during an iteration, the computations flow from the input layer, through any hidden layers, to the output layer. This architecture allows the network to learn to map input states to output states approximating the correct response to the input. For example, if the input states represent the symptoms presented by a medical patient, the network is able to produce output states representing an estimation of the correct diagnosis for those symptoms.

One of the hindrances to wider acceptance of neural networks is the fact that they function largely as black boxes. It is often difficult to understand why a specific set of inputs produced a particular output. This difficulty is a result of the fact that the network's 'knowledge' is encoded in the weights associated with a complex web of interconnections. It is desirable to find an explanation method for neural networks that is a way to explain a particular output in terms of the network inputs. For example, if a network is used to make loan decisions, it would be desirable to explain those decisions in terms of the input data describing the applicant. An explanation of this kind is required for negative loan decisions.

The black box problem is not as serious for two-layer feedforward networks, i.e., those without hidden layers. In these networks, the relationship between the inputs and outputs is straightforward. The magnitude and direction of the relationship between the states of an input PE and an output PE are given by the weight of the connection between the two PEs. Because the relationship between each input and output is fixed, these simple networks cannot capture variable relationships between input and outputs, such as non-monotonic relationships. Nor can they capture the interdependencies among inputs. That is to say, they cannot implement mappings in which the effect of some input on some output is dependent on the values of other inputs. They can learn only linear mappings, i.e., mappings where each output is proportional to a weighted sum of the inputs. Thus these networks are restricted to learning only a limited subset of the relationships that exist in the real world.

Networks with hidden layers are needed to learn nonlinear mappings, including non-monotonic relationships between inputs and outputs and interdependencies among inputs. Since there is no straightforward relationship between inputs and outputs in these networks, explanation is a difficult problem. One attempt at explanation uses sensitivity analysis. This technique involves changing the value of an input, iterating the network, and noting whether there is any meaningful change in the network's output. Using the medical domain as an example again, sensitivity analysis would involve changing one of the input symptoms and noting whether there is a change in the network's diagnostic output.

Explanation is also an issue in the field of expert systems. These systems are often compared to neural networks because the two technologies attempt to solve some of the same problems, namely: classification; prediction; and decision-making. Explanation is more straight-forward in expert systems than in neural networks, because the 'knowledge' in an expert system is more explicit in that it is contained in a set of if-then rules known as a rulebase. In addition to explanation, expert systems possess other desirable capabilities not found in neural networks of the prior art. These capabilities include the ability to determine when enough input information is present to make conclusions; to provide intelligent direction to the user's information acquisition; and to calculate confidence measures to accompany decisions.

Unfortunately, export systems lack the ability to learn from examples, which is the most appealing feature of neural networks. Although some attempts have been made to imbue expert systems with a learning capability, they still rely primarily on handcrafted rules as their source of 'knowledge'. Thus, building an expert system to solve a particular problem requires finding a human expert in the

problem domain, translating his knowledge into if-then rules, then debugging the rulebase.

Clearly, it is desirable to combine the learning ability of neural networks with the explanation and other capabilities of expert systems. One known attempt at such a combination involves an expert system which uses a neural network as its source of 'knowledge', and is thus able to take advantage of neural network learning. In addition to expert system capabilities such as explanation, this hybrid system includes an optional facility for converting the knowledge contained in the network into a set of rules. This system is described in the literature (see, for example, U.S. Patent 4,730,259).

Unfortunately, the techniques used in this hybrid system of the prior art have several important limitations. Most significantly, they are applicable only to perceptron networks, also known as linear discriminant networks. With the possible exception of the input layer, the PEs of these networks are limited to tertiary states, i.e., there are only three possible state values, corresponding roughly to TRUE, FALSE, and unknown. The result is that perceptron networks cannot compute mappings as precisely as continuous-state networks, i.e., networks whose states are not limited to a set of discrete values. Even more important than the loss of precision is the fact that perceptron networks cannot be trained with backpropagation learning, which is the most popular training method for networks with hidden layers.

The explanation techniques of the prior art hybrid system also have limitations. For example, explanations can be generated only for conclusions since there is no method for explaining why the system is leaning toward one output or another prior to a definitive conclusion. Explanations take the form of rules, with the conditions (inputs or hidden layer states) on the left-hand side of the rule serving as an explanation for the conclusion on the right-hand side. Those conditions included in the rule make a larger positive contribution to the conclusion than those omitted. However, no precise measurement of contributions is produced for either the included or omitted conditions.

The prior art hybrid system is also limited in the way in which it directs the user's information acquisition. The system chooses a single input whose value is unknown but important and asks the user to give a value for that input. However, the system provides no information about the relative importance of the other unknown inputs.

Summary of the Invention

Accordingly, the present invention is a system which allows expert system functionality, including explanation, to be added to feedforwerd neural networks. The invention overcomes the deficiencies in the prior art. In particular, the present invention overcomes the deficiencies in the existing technology for continuous-state, feedforward networks in that it can determine when enough input information is present to make conclusions; it can precisely explain its decisions and why some of those decisions become conclusions; it can calculate confidence measures to accompany its decisions; and it can compute the relative importance of the inputs with unknown values. The present invention is different from existing expert system technology because it can learn from examples, rather than relying on humans to give it knowledge in the form of rules. Explanation in the present invention is different from the existing technology for explanation in two-layer networks because the invention handles nonlinear relationships. The present invention is different from the hybrid system of the prior art because it overcomes the deficiencies in that system; for example, the invention operates on the powerful class of continuous-state feedforward networks.

Description of the Drawings

Figure 1 is a diagram of the major components, including inputs and outputs, according to the present invention;

Figure 2 is a flowchart of the operation of the inference module according to the present invention;

Figures 3A and 3B are flowcharts of the operation of the Conclusion Explanation according to the present invention;

Figure 4 is a flowchart of the operation of the Decision Explanation according to the present invention;

Figure 5 is a flowchart of the operation of the Intelligent Knowledge Acquisition module according to the present invention;

Figure 6 is a diagram of the neural network architecture and data flow therethrough according to the present invention for operation on a loan-scoring problem;

Figure 7 is a description of the thirteen input variables in the loan-scoring problem, along with sample values, in the operation illustrated in Figure 6;

Figure 8 is an example of output for Decision Explanation in the loan-scoring problem illustrated in Figures 6 and 7;

Figure 9 is an example of output for Conclusion Explanation in the loan-scoring problem illustrated in Figures 6 and 7;

Figure 10 is an example of output for Intelligent Knowledge Acquisition in the loan-scoring problem illustrated in Figures 6 and 7; and

Figure 11 is an example of output for Rule Generation in the loan-scoring problem illustrated in Figures 6 and 7.

Description of the Preferred Embodiment

To understand the present invention, it is important to understand the concept of input and output variables. A user of the present invention deals with these variables rather than dealing directly with the input and output states which underlie the variables. The decision-making process according to the present invention is viewed as mapping input variables to one or more output variables. Given values for some or all of the input variables, the present invention determines the values of the output variables. Those values correspond directly to the decisions made according to the present invention. For example, if there was an output variable which represented a loan decision, a value of TRUE might correspond to a decision to approve the loan.

The present invention has three types of input variables, namely: continuous; Boolean; and symbolic. A continuous input variable can have any numerical value and it is represented by a single PE whose state is equal to that value. If the value of the variable is unknown, the state is set to some default value, typically the mean value of the variable. A Boolean input variable can have a valve of either TRUE or FALSE. It is represented by a single PE with a state drawn from the set {-B,M,B}, corresponding to {FALSE, UNKNOWN, TRUE}. B is typically set at 1.0 and M is typically set to 0.0. A symbolic input variable draws its value from a finite set of symbols. An example is the variable 'region', which can have any value in the set {Northeast, Southeast, Northwest, Southwest}. Symbolic variables are represented using multiple PEs and a 1-out-of-n code. A symbolic variable with n possible values is represented by n PEs, each of which corresponds to a different value. If the value of the variable is known, the PE corresponding to the current value is given a state of B and the other PEs are given states of -B. If the value of the symbolic variable is unknown, all the PEs have states of M. As with Boolean variables, B is typically set to 1.0 and M is typically set to 0.0.

Output variables can be Boolean or symbolic. These types of variables are appropriate for the decision output of the present invention because they have categorical values, and decisions are inherently categorical. During operation in training mode, the network is presented with Boolean and symbolic training values, i.e., values corresponding to the correct decisions for the output variables. These training values are represented the same as Boolean or symbolic input values, using states in the set {-B,M,B}. However, the states produced in the output layer of a continuous-state feedforward network are, by definition, continuous and thus not restricted to {-B,M,B}. Therefore, the output states are interpreted as being approximations of Boolean and symbolic values.

When interpreting output states, a symbolic output variable is given the value corresponding to the underlying PE with the highest state. That PE is said to be the selected PE. A Boolean output variable is given a value according to a user-chosen Boolean decision threshold. If the underlying output state is greater than the threshold, the value of the variable is TRUE. Otherwise, the value of the variable is FALSE.

Referring now to the diagram of Figure 1, there are shown the essential elements of the present invention. These elements are referred to as components of the invention and they are divided into three categories: input, output, and internal components. One of the input components is the set of weights 9 that results from training the network with examples from the chosen problem domain. The weights 9 encode the 'knowledge' of the network about the domain. The method of training is independent of the present invention, so a user can choose from among the learning algorithms intended for feedforward networks. Backpropagation is the most widely used learning algorithm for the powerful class of continuous-state feedforward networks with hidden layers, on which the present invention can operate.

Another major input component is the input data 11, which specifies the values of the input variables for each case that is to be analyzed by the invention. The value of any input variable can be specified as UNKNOWN for a particular case, since the present invention in designed to intelligently handle unknown values.

Still another input component is the input variable statistics 13 which are applicable only if some of the input variables are continuous. These statistics describe the distribution of values for each continuous input variable. Specifically, the system requires either the mean and standard deviation for each such variable, or the minimum and maximum for each variable. In the latter case, the means are also needed if (1) they are used as the default state when the value of a continuous input variable is unknown; or (2) the explanation module 23 will be used for Rule Generation 31. The required distribution statistics can be computed from a data set, such as the set of training examples or the set of cases to be analyzed. Alternatively, the statistics can be estimated.

The diagram of Figure 1 illustrates that the present invention has three major internal components. The most fundamental of these components

is the Inference Module 15 which is responsible for three output components; namely: decision 17; conclusions 19; and confidence measures 21 (Decisiveness and Certainty). The operation of the Inference module is illustrated in the flowchart of Figure 2 with regard to a single output variable. The module and figure are discussed below.

The Inference module is operated whenever the values of input variables are added, deleted, or modified, as recorded in block 41 of Figure 2. Based on the new input values, the output states of the network are updated in block 43. Additionally, the module updates high and low bounds on those output states in block 45. These bounds are upper and lower limits on the possible range of each output state, given the known input values and any combination of values for some or all of the unknown input variables. Computing these bounds requires that assumptions be made about the possible values of continuous input variables. Details of these assumptions and the precise method for computing the bounds are explained following this overview of the Inference module.

The updated output states are used to determine the values of the output variables. Those values are the decisions 17 of the present invention. As previously explained, the method of determining output variable values or decisions from output states depends upon whether an output variable is Boolean or symbolic. If it is Boolean, then the underlying output state is compared to the user-chosen Boolean decision threshold in block 49. If the state is greater than the threshold, the decision is TRUE in block 51. Otherwise, the decision is FALSE in block 53. If an output variable is symbolic, the underlying PE with the highest output state is selected in block 59. In block 61, the decision is the symbol corresponding to the selected PE.

After updating the decision, the present invention determines whether it can declare the decision to be a conclusion 19. A conclusion can be reached when enough of the input values are known so that no combination of possible values for the unknown input variables could change the value of the output variable corresponding to the decision. For symbolic output variables, this conclusion condition is checked for in block 63. If the low bound on the state of the selected PE is greater than the high bound on the state of each unselected PE, then that selected PE is guaranteed to have the highest state given any combination of values for the unknown inputs. The conclusion condition is thus met, and the decision is declared to be a conclusion in block 67. Otherwise, the decision is said to be tentative in block 65.

For Boolean output variables, the method for checking the conclusion condition depends on the decision. If the decision is TRUE, the conclusion condition is checked in block 55. The condition is met when the low bound on the state of the underlying output PE is greater than the Boolean decision threshold, ensuring that the PE's state will exceed the threshold given any combination of values for the unknown inputs. If the decision is FALSE, the conclusion condition is checked in block 57. The condition is met when the high bound on the underlying output state is less than the decision threshold, ensuring that the state will be less than the threshold.

If all input values are known, the conclusion condition is guaranteed to be met. However, in real-world domains, it is often the case that some input values are unknown. In medical diagnosis, for example, important data may be missing because it is costly, time-consuming, or even risky to obtain. The conclusion-generating capability of the present invention overcomes the problem of missing data. The invention allows a user to know when he has enough input data to be sure of the computed decision.

The state bounds that are used to make conclusions are also used to compute the Certainty confidence measure 21 for a decision. Certainty is a measure of confidence that a conclusion for an output variable would be the same as the current decision for that variable. Certainty ranges from 0% to 100%, reaching the maximum value in block 71 only when a conclusion has been reached. Only then it is guaranteed that additional inputs won't change the decision. Before a conclusion is reached, Certainty must be computed in block 69. It is computed based on the distance between the output bounds required for the decision to become a conclusion and the actual output bounds.

For Boolean output variables, the Certainty percentage is computed as:

$(S(o,lo)-L)*100/(T-L)$ for TRUE decisions
$(U-S(o,hi))*100/(U-T)$ for FALSE decisions

where S(o, hi) and S(o, lo) are the high and low bounds on the state of the underlying output PE o, U and L are the upper and lower limits respectively on the output of the transfer function of o, and T is the Boolean decision threshold.

For symbolic output variables, the Certainty percentage is computed as:

$$[\{(S(o,lo)-S(o',hi))/(U-L)\} + 1]*100$$

where o is the selected output PE and o' is the unselected PE with the highest high bound.

A second confidence measure 21, Decisiveness, measures the strength of a decision and is computed in block 73. The maximum possible de-

cision strength is indicated by a Decisiveness value of 100%. A value near 0% indicates a weak decision. Decisiveness is computed based on the distance between output states and decision boundaries. Specifically, for Boolean output variables, the Decisiveness percentage equals:

(S(o)-T)*100/(U-T) for TRUE decisions
(T-S(o))*100/(T-L) for FALSE decisions

where S(o) is the state of the underlying output PE o.

For symbolic output variables, the Decisiveness percentage equals:

(S(o)-S(o'))*100/(U-L)

where o is the selected output PE and o' is the unselected PE with the highest state.

The medical domain provides an example of the usefulness of the confidence measures described above. If the present invention were used to make diagnostic decisions, those decisions could have life-and-death consequences. Before trusting a particular decision or acting upon it, a physician would want to know how much confidence he should have in that decision. Decisiveness will tell him how clearly the diagnostic decision is favored over alternative diagnoses. Certainty will tell him how likely it is that the decision would remain the same were he to gather additional information.

Note that the two confidence measures are only indirectly related. It is possible to have a high Certainty value but a low Decisiveness value or vise versa. For example, the former case occurs when a weak decision becomes a conclusion.

Turning now to the precise method for computing the bounds on output states, the method requires that assumptions be made about the possible values of continuous input variables. These assumptions are based on the input variable distribution statistics 13. If the user chooses to use the minimum and maximum statistics, the present invention assumes that the possible values of an unknown continuous input variable fall between its minimum and maximum. If the user chooses to use the mean and standard deviation instead, he must also specify a positive range value r, which applies to all continuous input variables. The invention assumes that the possible values of an unknown continuous variable i fall in the range defined by the following limits:

lower limit = Mean(i) - (r*Sd(i))
upper limit = Mean(i) + (r*Sd(i))

whore Mean(i) and Sd(i) are the mean value and standard deviation of variable i.

The bounds on the states of output PEs are computed using the following recursive equations for the state bounds of any non-input PE. If p is a non-input PE, its state bounds are computed as;

S(p,lo) = T(ϵ_sC(s,p,lo))
S(p,hi) = T(ϵ_sC(s,p,hi))

where T is the transfer function of p and C(s,p,hi) and C(s,p,lo) are the high and low bounds on the contributions made to the net-input of p by each of its sources s. Each s is either a hidden PE, input variable, or bias (special input PE whose state is always equal to 1.0).

Which of the three s is determines how the contribution bounds are computed. If s is a bias, then:

C(s,p,lo) = C(s,p,hi) = W(s,p)

where W(s,p) denotes the weight of the connection from s to p.

If s is a hidden PE, then the following equations are used:

If W(s,p)>0 then

C(s,p,lo) = W(s,p)*S(s,lo)
C(s,p,hi) = W(s,p)*S(s,hi)

Otherwise

C(s,p,lo) = W(s,p)*S(s,hi)
C(s,p,hi) = W(s,p)*S(s,lo)

If s is an input variable, then the calculation of contribution bounds depends on the data type of s and whether s is known or unknown. If s is Boolean or continuous and known to have a value v, then:

C(s,p,lo) = C(s,p,hi) = v*W(i,p)

where i is the PE underlying s.

If s is Boolean or continuous and its value is unknown, then the following equations are used:

If W(i,p)>0 then

C(s,p,lo) = W(i,p)*Min(s)
C(s,p,hi) = W(i,p)*Max(s)

Otherwise

C(s,p,lo) = W(i,p)*Max(s)
C(s,p,hi) = W(i,p)*Min(s)

where Max(s) and Min(s) are the maximum and minimum possible values of s. If s is Boolean, Max(s) is B and Min(a) is -B, where B and -B are the

values used to represent TRUE and FALSE respectively. If s is continuous, Max(s) and Min(s) are the upper and lower assumed limits on s, as derived from the distribution statistics.

If s is symbolic and known to have a value corresponding to PE v in the underlying choice group, then:

$$C(s,p,lo) = C(s,p,hi) = C(s,p,v)$$

where C(s,p,v) is the contribution s makes to the net-input of p when the value of s corresponds to v. Specifically,

$$C(s,p,v) = (W(v,p) - \epsilon_{i<>v}W(i,p))^{*}B$$

where i iterates over the PEs in the underlying choice group.

If s is symbolic and its value is unknown, then:

$$C(s,p,lo) = Min(C(s,p,i))$$
$$C(s,p,hi) = Max(C(s,p,i))$$

where Max and Min are computed over all PEs i in the underlying choice group.

An example problem is now used to demonstrate the operation of the Inference module described directly above. The example involves making decisions on personal loan applications, based on individual applicant data. The neural network architecture and data flow used for this problem are illustrated in Figure 6. The network is simulated using computer software, although the present invention is equally applicable to a network implemented in hardware. The input layer 203 of the network contains thirteen PEs, one for each of thirteen continuous input variables. The input layer is fully connected to a single hidden layer 205 containing three PEs, which in turn is fully connected to an output layer 207 consisting of a single PE. The output PE represents a Boolean variable whose value corresponds to the loan decision 209. A TRUE value corresponds to a decision to approve the loan, and a FALSE value corresponds to a decision to decline. The output and hidden PEs have a conventional sigmoid transfer function, with output in the range [-1.0,1.0].

Note that the network in Figure 6 is just one example of the type of neural network on which the present invention can operate. In general, the invention can operate on a feedforward network with any number of hidden layers (including zero), any number of PEs in the hidden, input, and output layers, and an arbitrary transfer function for a PE (different PEs can have different transfer functions).

The particular network is trained using backpropagation learning, which modifies the weights 211. The backpropagation learning algorithm, which is independent of the present invention, involves a backward data flow in addition to the feedforward flow depicted in Figure 6. The network of Figure 6 is trained on data 201 which represents examples of previous applicants whose loan outcomes are known. Each example is assigned a training value of either Approve (1.0) or Decline (-1.0), depending on whether the applicant defaulted on the loan. The applicant is described by the values of the thirteen input variables, which are explained in the table of Figure 7. The table also contains the input values for an actual applicant in the test database.

Using the network in Figure 6, the present invention is able to make a conclusion about this sample applicant, despite two missing input values in the data described in Figure 7. The concluded decision is to approve the loan. This decision corresponds to a TRUE value for the output variable and is a correct decision in that the applicant satisfactorily repaid the actual loan. Since a conclusion was reached, Certainty is 100%. Decisiveness is computed to be 13.4%, using the formula described above for Decisiveness given a TRUE decision (Decisiveness was usually below 40% for examples in the test database). In order to further demonstrate, assume that only those values in Figure 7 which came from a credit report (i.e., values for Active Accounts, New Accounts, # of Inquiries, Public Record Items, Time in Credit File, and the three Overdue variables) are known. Now there are too many unknown inputs to allow a conclusion, so the present invention produces a tentative decision instead. The decision is to approve, with Decisiveness of 10.1% and Certainty or 47.4%. These values are computed using the Decisiveness and Certainty formulas for TRUE decisions.

Referring again to Figure 1, the second internal component is the Explanation module 23. This module produces two types of explanation. One type explains any decision, whether tentative or conclusive, by computing the contribution each known input value makes to that decision. This type is Decision Explanation 25 and is based on output states. The other type, Conclusion Explanation 27, examines output bounds to explain how a conclusion was reached. Conclusion Explanation computes the contributions made by known inputs toward reaching the conclusion condition, and also determines a minimal subset of those inputs that is sufficient to support the conclusion.

Note that both types of contributions are computed in the context of the current set of input values. Thus the contribution made by a given input variable to a given decision or conclusion is dependent not just on the variable's own value, but also on the values (or lack thereof) of the other input variables. Thus, both types of explanation capture the interdependency among inputs that is

found in feedforward networks with hidden layers.

Conclusion Explanation 27 computes a contribution for each known input variable that is equal to the decrease in the strength of the conclusion condition when the value of the variable is assumed to be unknown. That strength is measured as the distance between the bounds representing the conclusion condition and the actual bounds on the states underlying the output variable for which the conclusion has been reached. The operation of Conclusion Explanation is illustrated in the flowchart of Figure 3A with regard to a single output variable. The module and figure are discussed below.

In block 81 of Figure 3A, the present invention checks to see if a conclusion has been reached for the output variable for which explanation is desired. If no conclusion has been reached, Conclusion Explanation is not possible in block 83. Otherwise, Conclusion Explanation proceeds in block 85 by examining the first or next input variable whose value is known. The value of that variable is temporarily assumed to be unknown, and the bounds on the output states are temporarily adjusted in block 87 to reflect that assumption. The adjusted and true (before adjustment) output bounds are compared to determine the contribution of the input variable being examined.

The method of computing the contribution is dependent on whether the output variable is Boolean or symbolic in block 89. If the output variable is Boolean, the contribution is further dependent on whether the decision for that variable is TRUE or FALSE in block 91. If the decision is TRUE, the conclusion condition is based on the low bound on the state of the PE underlying the Boolean output variable, and thus the contribution in block 93 is equal to the true low bound minus the adjusted low bound on the underlying state. If the decision is FALSE, the conclusion condition is based on the high bound, and thus the contribution in block 95 is the adjusted high bound minus the true high bound on the underlying state. Formally, the contributions equal:

S(o,lo)-S' (o,lo) for TRUE decisions
S'(o,hi)-S(o,hi) for FALSE decisions

where o is the underlying output PE, S signifies the true bounds, and S' signifies the adjusted bounds.

If the output variable is symbolic, the conclusion condition is based on the gap between the low bound on the state of the selected PE and the highest high bound among the unselected PEs. Thus, for symbolic output variables, the contribution in block 97 is equal to the gap given the true bounds minus the gap given the adjusted bounds. Formally, the contribution equals:

(S(o,lo)-S(o',hi))-(S'(o,lo)-S'(o'',hi))

where o is the selected output PE, o' is the unselected PE with the highest true high bound, and o'' is the unselected PE with the highest adjusted high bound.

Note that all contributions computed by Conclusion Explanation will be non-negative. This results from the combination of these facts: 1. the contribution of an input variable is computed relative to that variable being unknown; 2. when the value of an input variable becomes known, the bounds on each output state either stay the same or become narrower, where narrower bounds mean that the high bound has decreased and/or the low bound has increased; 3. Narrower output bounds can strengthen, but not weaken, the conclusion condition; and 4. the contributions measure the strengthening of the conclusion condition.

After computing the contribution for a particular input variable, the present invention checks in block 99 to see whether all known input variables have been examined. If not, the next known variable is examined, Otherwise, Conclusion Explanation proceeds to block 101, where the contributions for all known input variables are multiplied by a scaling constant, determined such that the largest scaled contribution is 100.0. The constant is calculated by dividing 100.0 by the maximum of the unscaled contributions. In block 103, the scaled contributions, along with the corresponding variable names and values, are produced as output. The output is ordered numerically by contribution.

Conclusion Explanation further explains a conclusion by finding the minimal premises, a subset of the known input values that, by themselves, are sufficient for reaching the conclusion. Known inputs not in the minimal premises could be unknown and the conclusion would still hold. The conclusion and minimal premises can be viewed together as a rule which states a set of sufficient conditions for reaching the conclusion, independent of the case currently being analyzed.

The operation of Conclusion Explanation in finding the minimal premises is illustrated in Figure 3B. The first step is to set k to the number of known input variables and to initialize the count c to zero in block 111. Each time blocK 113 is reached, c is incremented by 1. The first time through block 115, the present invention chooses the known input variable with the smallest contribution. It assumes the value of that variable to be unknown in block 117. In block 119, it checks to see if the conclusion condition, as determined by the Inference module, still holds in light of that assumption. If the conclusion is still supported, the known input variable with the second smallest con-

tribution is chosen in a return to block 115. The value of this variable is also assumed to be unknown. This process repeats, with c counting the number of known input variables assumed to be unknown. When the conclusion no longer holds in block 119, the minimal premises are determined in block 121. The minimal premises are determined to consist of the k-c+1 inputs with the largest contributions.

Figure 9 contains an example of the output of Conclusion Explanation, using the sample loan applicant described in Figure 7 and the network described in Figure 6. For each known input variable, the output lists the contribution and indicates whether the variable belongs to the minimal premises. The contributions were computed using the formula described above for contributions given a TRUE decision. The minimal premises are distinguished in the output with the heading Sufficient. Known input variables that are not among the minimal premises are listed under Additional. Figure 9 shows that the minimal premises, in this case, include all known input variables except for Time in Credit File. It also shows that the values of 30-Days Overdue Now and # of Inquiries were the most important factors in reaching the Approve conclusion.

The second type of explanation, Decision Explanation 25, measures the contribution made by each known input variable to a decision. This type of explanation is best explained by contrasting it to Conclusion Explanation 27. Both types of explanation measure the contribution made by each known input variable by examining the change in output when the value of the variable is assumed to be unknown. However, Conclusion Explanation examines changes in output bounds, whereas Decision Explanation examines changes in output states. This difference is motivated by the fact that Conclusion Explanation measures contributions to the conclusion condition, which is based on output bounds, while Decision Explanation measures contributions to a decision, which is based on output states.

An additional difference concerns the sign of the contributions. As explained earlier, contributions in Conclusion Explanation cannot be negative, because knowing the value of an input variable can only strengthen the conclusion condition or leave it unchanged, relative to not knowing the value. However, knowing the value of an input variable can weaken as well as strengthen a decision, as indicated by a decrease or increase in the Decisiveness confidence measure. Thus, contributions in Decision Explanation can be negative or positive. As a matter of fact, a particular value for an input variable can make a positive contribution to a particular decision in one input context, yet make a negative contribution to that same decision in another context. This is due to the interdependency among inputs that is found in feedforward networks with hidden layers.

The operation of Decision Explanation is illustrated in the flowchart of Figure 4 with regard to a single output variable. The module and figure are explained as follows: in block 131, the present invention examines the first or next input variable whose value is known. The value of that variable is temporarily assumed to be unknown, and the output states are temporarily adjusted in block 133 to reflect that assumption. The adjusted and true (before adjustment) output states are compared to determine the contribution of the input variable being examined.

The method of computing the contribution is dependent on whether the output variable is Boolean or symbolic in block 135. If the output variable is Boolean, the contribution is further dependent on whether the decision for that variable is TRUE or FALSE in block 137. If the decision is TRUE, the contribution in block 139 is equal to the true state minus the adjusted state of the PE underlying the Boolean output variable. If the decision is FALSE, the contribution in block 141 is the adjusted output state minus the true output state. Formally, the contributions equal:

$S(o)-S'(o)$ for TRUE decisions
$S'(o)-S(o)$ for FALSE decisions

where o is the underlying output PE, S signifies the true state, and S' signifies the adjusted state.

If the output variable is symbolic, the contribution in block 143 is based on the gap between the state of the selected PE and the highest state among the unselected PEs. The contribution is equal to the gap given the true states minus the gap given the adjusted states. Formally, the contribution equals:

$$(S(o)-S(o'))-(S'(o)-S'(o''))$$

where o is the selected output PE, o' is the unselected PE with the highest true state, and o'' is the unselected PE with the highest adjusted state.

After computing the contribution for a particular input variable, the present invention checks in block 145 to see whether all known input variables have been examined. If not, the next known variable is examined. Otherwise, Decision Explanation proceeds to block 147, where the contributions for all known input variables are multiplied by a scaling constant, determined such that the largest absolute value of any contribution is 100.0. The constant is calculated by dividing 100.0 by the maximum absolute value of the unscaled contributions.

In block 149, the scaled contributions, along with the corresponding variable names and values, are produced as output. The output is ordered numerically by contribution. In addition, the output distinguishes three groups of known input variables: 1. those which make insignificant contributions to the decision, i.e., contributions whose absolute values fall below a user-chosen significance threshold; 2. those which significantly support the decision, i.e., those with positive contributions at or above the threshold; and 3. those which significantly weaken the decision, i.e., those with negative contributions whose absolute values are at or above the threshold.

Figure 8 contains an example of the output of Decision Explanation, again using the loan applicant described in Figure 7 and the network described in Figure 6. The output was produced using a significance threshold of 20.0. The contributions listed in the output were computed using the formula described above for contributions given a TRUE decision. The output shows that there were four factors significantly supporting the Approve decision. The largest such factor was the value of New Accounts. The other three factors involve the Overdue variables. The only factor significantly weighing against approval was the value of Time in Credit File. Note in Figure 9 that Time in Credit File was also the least supportive input variable as indicated by Conclusion Explanation. It made the smallest contribution and was the only known input variable not in the minimal premises. However, in general, there is only an indirect relationship between a variable's ranking according to the two types of explanation, since each measures a different property.

Determining contributions in Decision Explanation requires computing adjusted output states, based on the assumption that a known input variable is unknown. However, computing the adjusted states does not require network iterations. This contrasts with conventional sensitivity analysis, where the effect of changing an input is determined by iterating the network after making the change. The present invention includes a method for finding the modified output resulting from a change in input, while avoiding the computational expense of a network iteration.

Computation of a network iteration involves all the weights in the network, but in accordance with the present invention, a simpler partial weights method is used which involves only a subset of the weights. This is important because the computer time required for the computation is roughly proportional to the number of weights used. Specifically, the partial weights method, according to the present invention, uses only the weights associated with connections that lie in a path between one of the input PEs whose state is being changed and one of the output PEs for which state change is being measured.

Exactly which connections lie in such a path depends on the number of hidden layers and on the connectivity of the network. In order to provide a specific example, reference is made to the network in Figure 6. Suppose it is necessary to find the modified output state of this network that results from changing the state of input PE i from v to v'. The partial weights method uses only the weights for the connections from i to each of the three hidden PEs and the connections from each hidden PE to the output PE.

The first step in the partial weights computation for the network in Figure 6 is to calculate the resulting net-input of each of the three hidden PEs. The resulting net-input $I'(h)$ for each hidden node h is equal to $I(h)+((v'-v)^*W(i,h))$, where $W(i,h)$ is the weight of the connection from i to h and $I(h)$ is the original net-input for h. The next step is to compute the resulting state for each of the hidden PEs. $S'(h)$, the new state of h, is equal to $T(I'(h))$, where T is the conventional sigmoid transfer function used by the network of Figure 6. The modified output state is then computed by applying T to the weighted sum of each $S'(h)$.

Using the partial weights method, according to the present invention, each of the decision contributions in Figure 8 were computed with only 6 weights. Without the method, each contribution would have required computing a network iteration, which uses 42 weights. The partial weights method is also used to compute the changes in decisions and Decisiveness that result when the user adds, modifies, or deletes input values.

In addition to explaining decisions and conclusions, the Explanation module 23 has the ability to do Rule Generation 31. This process produces a set of if-then rules, each of which consists of some number of possible input values and a decision. A rule is interpreted to mean that the decision would be supported as a conclusion if the input values were known to be true. These rules are intended to approximately represent the knowledge implicit in the weights of the network. By making the relationship between input values and decisions explicit, the knowledge in the network is made explicit. The conversion from weights to rules results in some loss of precision, but the rules are intended only as an aid to a user who wishes to analyze the knowledge contained in the network, and are not a requirement for any of the other capabilities of the present invention.

A key to Rule Generation is the minimal premises 29 described earlier. The input values in a rule constitute a set of minimal premises for reaching the conclusion in the rule. In other words, Rule

Generation constructs rules such that if any of the input values in the rule were removed, the conclusion of the rule would no longer hold. Rule Generation constructs rules by producing sets of minimal premises to support hypothetical conclusions, where the premises consist of hypothetical (but possible) input values. Specifically, rules are generated for an output variable by finding sets of hypothetical minimal premises that can support one of the possible decisions for that variable as a conclusion. Each unique set of minimal premises found results in a rule. All possible sets are found and thus all possible rules are generated, restricted only by the user's option to specify a maximum number of premises per rule.

All possible sets of minimal premises are found by doing a search of input space. Specifically, Rule Generation uses a conventional depth-first tree search. In the context of the present invention, the paths through the simulated tree correspond to possible combinations of input values. Each level in the tree corresponds to a different input variable. Each node in the tree has n + 1 branches, where n is the number of possible values for the input variable corresponding to the level at which the node exists. One branch corresponds to each possible value and the extra branch corresponds to the value being unknown. For Boolean input variables, the possible values are TRUE and FALSE, so n is 2. For symbolic variables, n is the number of possible symbols. For continuous variables, there are an infinite number of possible values, so a representative sample of the values must be chosen. The solution in the present invention is to choose three values representative of different regions within the value range of a given continuous input variable. The specific method for choosing these values is explained later.

Searching along a particular path in the tree is interpreted as choosing the value (possibly UNKNOWN) corresponding to each branch in the path. Thus, when the search reaches m levels down in the tree, the values of m input variables have been either set or said to be unknown. The search follows each possible path until the set values result in a conclusion or the bottom of the tree is reached. If the bottom is reached without a conclusion, then that path produces no rule. However, if a conclusion is reached, the known values set in that path, along with the conclusion, constitute a candidate rule. The candidate rule will be chosen only if its premises are minimal, i.e., if all of the known values set in the path are required to support the conclusion. Rule Generation checks to see if any of the known values can be assumed to be unknown without a resulting loss of the conclusion. If so, the premises are not minimal, and the candidate rule is thrown away. Otherwise, the candidate rule is included in the output of Rule Generation.

Turning now to the precise method for choosing the three sample values for each continuous input variables, Rule Generation chooses a value from the upper, lower, and middle regions of the variable's value range. The chosen values are referred to as High, Low, and Medium, and are computed from the input variable statistics 13. The Medium value is equal to the mean of the input variable. The values for High and Low depend on which statistics are used. If the minimum and maximum statistics are used, the following values are chosen for continuous input variable i:

Low = (3 Min(i) + Max(i))/4
High = (3 May(i) + Min(i))/4

where Min(i) and Max(i) are the minimum and maximum values of i. The chosen values correspond to points one-quarter and three-quarters of the way through the interval [Min(i), Max(i)].

If the mean and standard deviation statistics are used, the following values are chosen:

Low = Mean(i) - (z*Sd(i))
High = Mean(i) + (z*Sd(i))

where Mean(i) and Sd(i) are the mean value and standard deviation of variable i, and z is a positive value chosen by the user that applies to all continuous input variables.

Figure 11 contains a typical rule produced by Rule Generation, using the method described above. The loan-scoring problem is again used as an example. Since this problem involves continuous input variables, it provides a good example of the present invention's method for using continuous variables as rule premises, i.e., the method of using High, Low, and Medium values. The rule in Figure 11 specifies six input values which are sufficient for supporting an Approve conclusion. If an applicant has those six values, he can be assured of approval, regardless of his values for the remaining seven input variables.

Referring again to Figure 1, the third internal component is the Intelligent Knowledge Acquisition module 33. The term 'intelligent knowledge acquisition' refers to a systematic method for determining what knowledge is most profitable to pursue. In the context of the present invention, this involves estimating which unknown input variables will have the greatest impact on a particular output variable if their values become known. By pursuing values for the unknown input variables with the greatest influence, the user can decrease the number of additional inputs needed to reach a conclusion. This is especially important in domains, such as medical

diagnosis, where there can be a significant cost, risk, or time delay associated with gathering information.

The operation of the Intelligent Knowledge Acquisition (IKA) module is illustrated in the flowchart of Figure 5. The module measures the potential influence of each unknown input variable 33 on a user-chosen output variable. In block 161 of Figure 5, the IKA module examines the first or next input variable whose value is unknown. Test values are determined for the variable in block 163. If the variable is Boolean or symbolic, each possible value of the variable is used as a test value. If the variable is continuous, two test values are chosen. They are computed using the same formulas by which High and Low values are chosen for continuous input variables in Rule Generation 31.

After the test values for the input variable are determined, an influence total is initialized to zero in block 165. The first or next test value is chosen in block 167 and the input variable is temporarily assumed to have that value. The module computes the contribution made by that input value to the decision for the user-chosen output variable. The contribution is computed using the same formulas by which Decision Explanation 25 computes contributions. The absolute value of the computed contribution is added to the influence total in block 169. If all test values have been used in block 171, an influence measure for the input variable is computed in block 173 by dividing the influence total by the number of test values. The influence of an unknown variable is thus computed to be the mean absolute value of the contributions made by the variable's test values.

After computing the influence of a variable, the IKA module checks to see, in block 175, if all unknown input variables have been examined. If not, the next unknown variable is examined. Otherwise, the module proceeds to block 177, where the influence measures for all unknown input variables are multiplied by a scaling constant, determined such that the largest measure is 100.0. The constant is calculated by dividing 100.0 by the maximum of the unscaled measures. In block 179, the scaled influence measures, along with the corresponding variable names, are produced as output. The output is ordered numerically by influence measure.

Figure 10 contains an example of the output of the IKA module, again using the loan applicant described in Figure 7 and the network described in Figure 6. As in an earlier example, this example assumes that we know only those values listed in Figure 7 that came from a credit report. That leaves five unknown input variables. The output in Figure 10 shows the influence measures computed for those five variables, using the method illustrated in the flowchart of Figure 5. The measures indicate that knowing the value of Bankcard Accounts or Time at Job would be most valuable for reaching a conclusion, whereas knowing the value of Time at Address would be least valuable.

Note that the influence measures computed by the IKA module are dependent on the current set of input values, as is the case with the contributions computed by the Explanation module 23. Thus, an unknown input variable may be important in some situations, as indicated by a large influence measure, and unimportant in others.

Claims

1. A method of operating an expert system based on a trained continuous-state feedforward neural network, where the network includes a layer of input processing elements (PEs), a layer of output PEs, and any number of layers of hidden PEs, where the input PEs are representative of input variables and the output PEs are representative of output variables, the method comprising the steps of:

   determining a decision for each output variable based on the states of the underlying PEs;

   determining a conclusion for each output variable when, as inputs to the network become known, the decision becomes irreversible;

   computing confidence measures for each decision;

   explaining each decision by computing the contribution made toward the decision by each input variable with a known value;

   explaining each conclusion by computing the contribution made toward reaching the conclusion by each input variable with a known value;

   determining the potential influence on a decision of each input variable with an unknown value; and

   converting the knowledge implicit in the neural network into an explicit set of if-then rules.

2. A method for computing high and low bounds on the states of hidden and output PEs a feedforward neural network, wherein these bounds serve as limits on the possible range of each such state, given known input states and any combination of possible states for some or all of the input PEs whose states are currently unknown, the method comprising the steps of:

   determining the upper and lower bounds on the values of continuous input variables,

using the input variable distribution statistics; and

computing the bounds on the states of the output PEs recursively from the bounds on the input variables and the bounds on the states of the hidden PEs.

3. A method according to claim 2 for determining when a conclusion can be made regarding the value of an output variable in a feedforward neural network with continuous-state PEs, despite some number of unknown input states, the method comprising the steps of:

determining the high and low bounds on the output states;

using the high and low bounds on the output states to determine if the tentative decision for the output variable can ever change; and

producing a conclusion if the tentative decision cannot change.

4. A method for computing a confidence measure of decisiveness for a decision corresponding to the value of an output variable in a feedforward neural network, the method comprising the steps of:

determining the network output states given a set of known input values; and

comparing the output states with the selected threshold or other decision criteria and quantitatively measuring the closeness of the output states to the decision criteria.

5. A method according to claim 2 for computing a confidence measure of certainty for a decision corresponding to the value of an output variable in a feedforward neural network, the method comprising the steps of:

determining the bounds on the states of the output PEs;

comparing the output bounds with the selected threshold or other decision criteria and quantitatively measuring the closeness of the bounds to the decision criteria to determine the probability that the output decision will not be reversed.

6. A method of explaining the reasons for a neural network output decision, the method comprising the steps of:

determining the network output states on a set of known input values;

comparing the output states with the decision criteria and determining an output decision;

sequentially converting each known input value to an unknown condition;

for each input value which is converted from known to unknown, measuring the change in the output state relative to the decision criteria; and

ranking quantitatively the importance of each known input value in determining the output decision.

7. A method of explanation according to claim 3 for measuring the contribution made toward reaching the conclusion condition by each input variable with a known value in a feedforward neural network, the method comprising the steps of:

determining the upper and lower output state bounds;

sequentially converting each known input value to the unknown condition;

for each input value which is converted from known to unknown, measuring the change in the upper and lower output bounds relative to the decision criteria; and

ranking quantitatively the importance of each known input value in reaching the conclusion condition.

8. A method according to claim 3 for computing a minimal subset of current input values that is sufficient to support the conclusion condition in a continuous-state feedforward neural network, the method comprising the steps of:

evaluating the relative changes in the output bounds for each known input value; and

selecting the minimum set of the most important inputs which are sufficient to reach the conclusion condition.

9. A method of measuring the potential influence an unknown input variable may have on the decision corresponding to the output of a feedforward neural network if the value of the variable becomes known, the method comprising the steps of:

for each unknown input value, sequentially determining the change in the output state as each input is varied over its possible value range; and

comparing quantitatively the changes in output states relative to the decision criteria for each unknown input.

10. A method according to claim 3 for using the conclusion generating process to translate the knowledge implicit in a continuous-state feedforward neural network into an explicit set of if-then rules supporting possible conclusions, the method comprising the steps of:

determining a minimum set of input values

which will result in reaching the conclusion condition, where the set of input values then constitute a rule for the particular conclusion; and

searching input values for the set of all possible rules which support any conclusion for a particular output variable.

11. A method of using the values of continuous input variables in the rules according to claim 10 wherein specific values are chosen to be representative of the upper, lower, and middle statistical regions of the variable's value range, and those values are examined for possible inclusion in one or more rules.

12. A method according to claim 6 for using a subset of the weights of a feedforward neural network to compute the modified output states that result from an actual change or an assumed change in input values, the method comprising the additional steps of using the weights associated with connections that lie in a path between one of the input PEs whose state is being changed and one of the output PEs for which a modified state is being computed to compute the modified states.

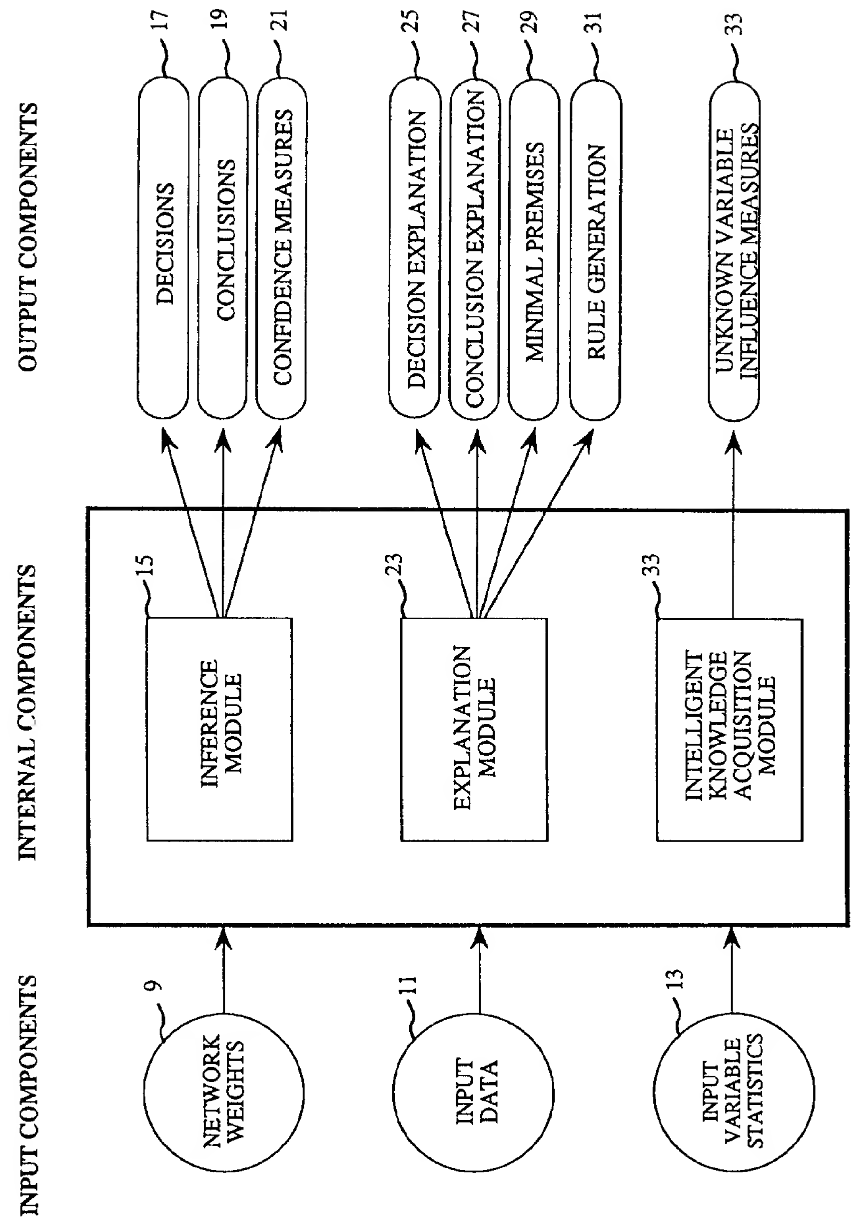
INPUT COMPONENTS
INTERNAL COMPONENTS
OUTPUT COMPONENTS
NETWORK WEIGHTS
9
INPUT DATA
11
INPUT VARIABLE STATISTICS
13
INFERENCE MODULE
15
EXPLANATION MODULE
23
INTELLIGENT KNOWLEDGE ACQUISITION MODULE
33
DECISIONS
17
CONCLUSIONS
19
CONFIDENCE MEASURES
21
DECISION EXPLANATION
25
CONCLUSION EXPLANATION
27
MINIMAL PREMISES
29
RULE GENERATION
31
UNKNOWN VARIABLE INFLUENCE MEASURES
33

Figure 1

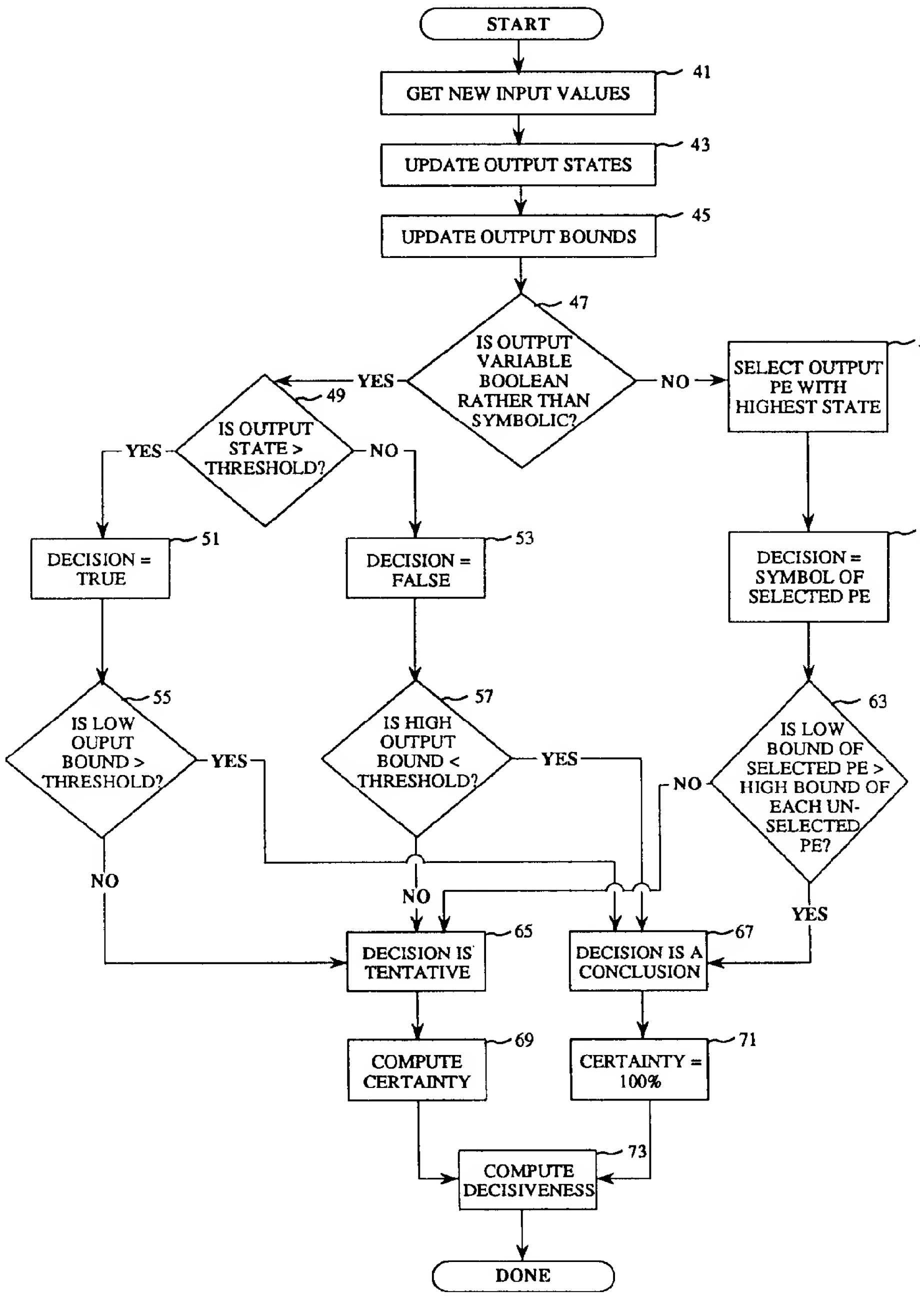
START
GET NEW INPUT VALUES
41
UPDATE OUTPUT STATES
43
UPDATE OUTPUT BOUNDS
45
IS OUTPUT VARIABLE BOOLEAN RATHER THAN SYMBOLIC?
47
YES
NO
SELECT OUTPUT PE WITH HIGHEST STATE
59
IS OUTPUT STATE > THRESHOLD?
49
YES
NO
DECISION = TRUE
51
DECISION = FALSE
53
DECISION = SYMBOL OF SELECTED PE
61
IS LOW OUPUT BOUND > THRESHOLD?
55
YES
NO
IS HIGH OUTPUT BOUND < THRESHOLD?
57
YES
NO
IS LOW BOUND OF SELECTED PE > HIGH BOUND OF EACH UN-SELECTED PE?
63
NO
YES
DECISION IS TENTATIVE
65
DECISION IS A CONCLUSION
67
COMPUTE CERTAINTY
69
CERTAINTY = 100%
71
COMPUTE DECISIVENESS
73
DONE

Figure 2

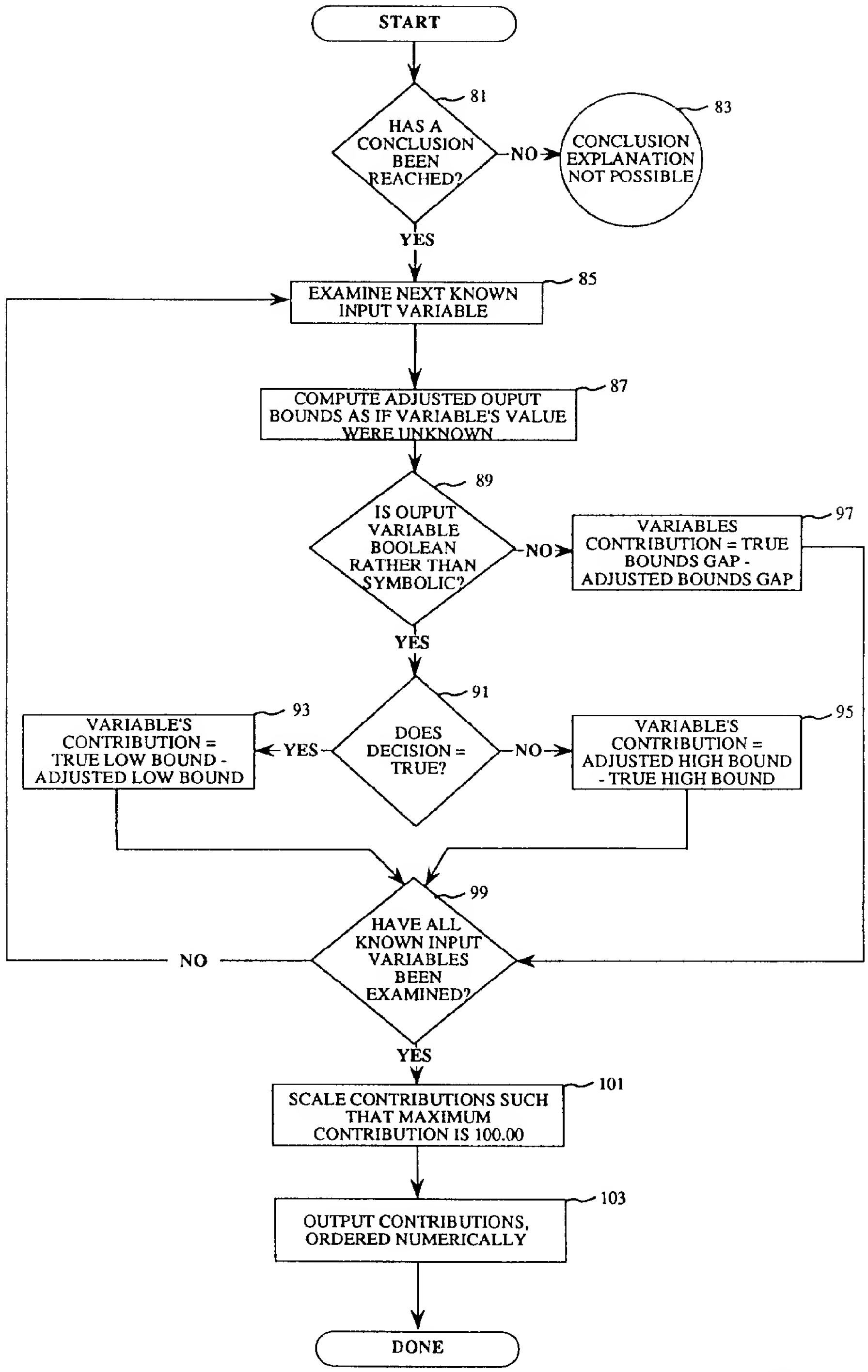
START
HAS A CONCLUSION BEEN REACHED?
81
NO
CONCLUSION EXPLANATION NOT POSSIBLE
83
YES
EXAMINE NEXT KNOWN INPUT VARIABLE
85
COMPUTE ADJUSTED OUPUT BOUNDS AS IF VARIABLE'S VALUE WERE UNKNOWN
87
IS OUPUT VARIABLE BOOLEAN RATHER THAN SYMBOLIC?
89
NO
VARIABLES CONTRIBUTION = TRUE BOUNDS GAP - ADJUSTED BOUNDS GAP
97
YES
DOES DECISION = TRUE?
91
YES
VARIABLE'S CONTRIBUTION = TRUE LOW BOUND - ADJUSTED LOW BOUND
93
NO
VARIABLE'S CONTRIBUTION = ADJUSTED HIGH BOUND - TRUE HIGH BOUND
95
HAVE ALL KNOWN INPUT VARIABLES BEEN EXAMINED?
99
NO
YES
SCALE CONTRIBUTIONS SUCH THAT MAXIMUM CONTRIBUTION IS 100.00
101
OUTPUT CONTRIBUTIONS, ORDERED NUMERICALLY
103
DONE

Figure 3A

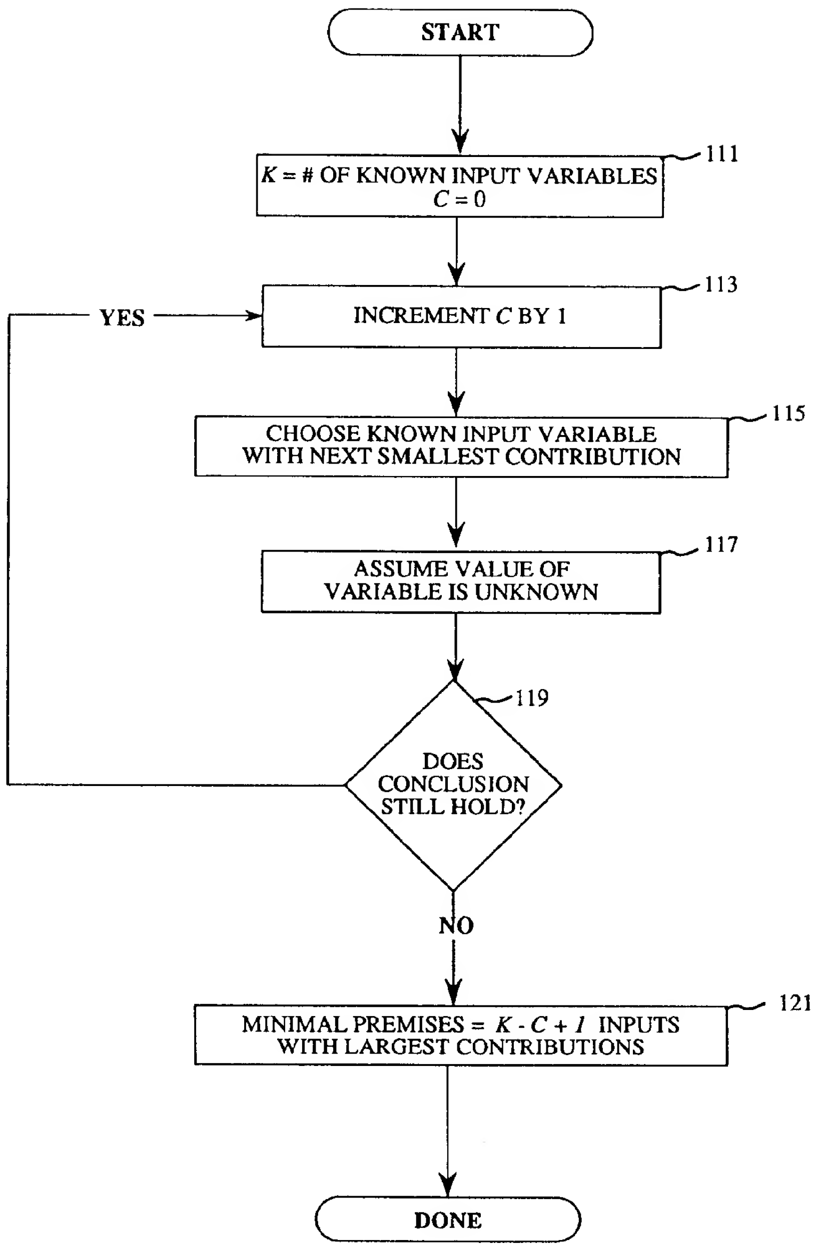
START
K = # OF KNOWN INPUT VARIABLES
$C = 0$
111
INCREMENT C BY 1
113
YES
CHOOSE KNOWN INPUT VARIABLE
WITH NEXT SMALLEST CONTRIBUTION
115
ASSUME VALUE OF
VARIABLE IS UNKNOWN
117
DOES
CONCLUSION
STILL HOLD?
119
NO
MINIMAL PREMISES = $K - C + 1$ INPUTS
WITH LARGEST CONTRIBUTIONS
121
DONE

Figure 3B

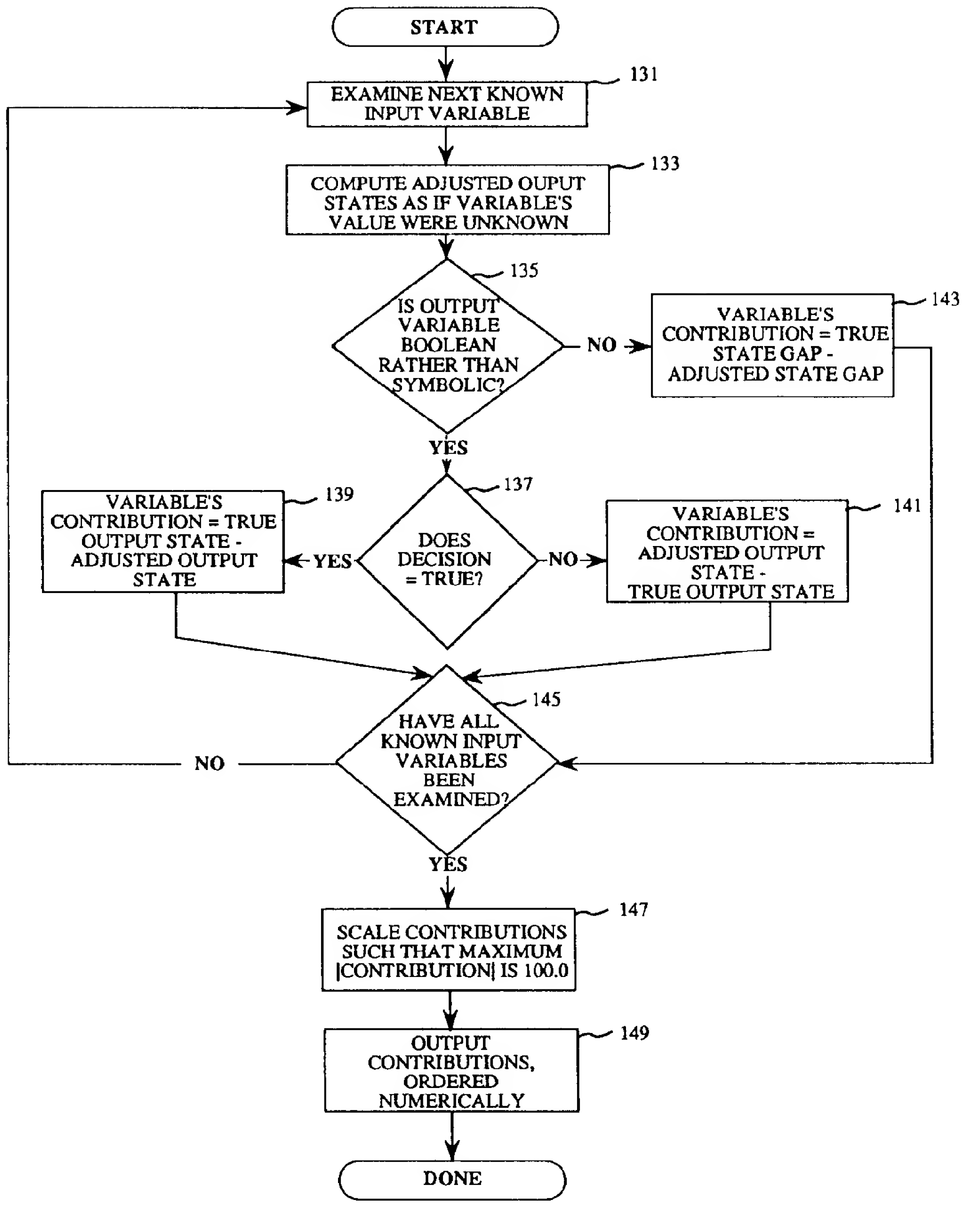
START
EXAMINE NEXT KNOWN INPUT VARIABLE
131
COMPUTE ADJUSTED OUPUT STATES AS IF VARIABLE'S VALUE WERE UNKNOWN
133
IS OUTPUT VARIABLE BOOLEAN RATHER THAN SYMBOLIC?
135
NO
VARIABLE'S CONTRIBUTION = TRUE STATE GAP - ADJUSTED STATE GAP
143
YES
DOES DECISION = TRUE?
137
YES
VARIABLE'S CONTRIBUTION = TRUE OUTPUT STATE - ADJUSTED OUTPUT STATE
139
NO
VARIABLE'S CONTRIBUTION = ADJUSTED OUTPUT STATE - TRUE OUTPUT STATE
141
HAVE ALL KNOWN INPUT VARIABLES BEEN EXAMINED?
145
NO
YES
SCALE CONTRIBUTIONS SUCH THAT MAXIMUM |CONTRIBUTION| IS 100.0
147
OUTPUT CONTRIBUTIONS, ORDERED NUMERICALLY
149
DONE

Figure 4

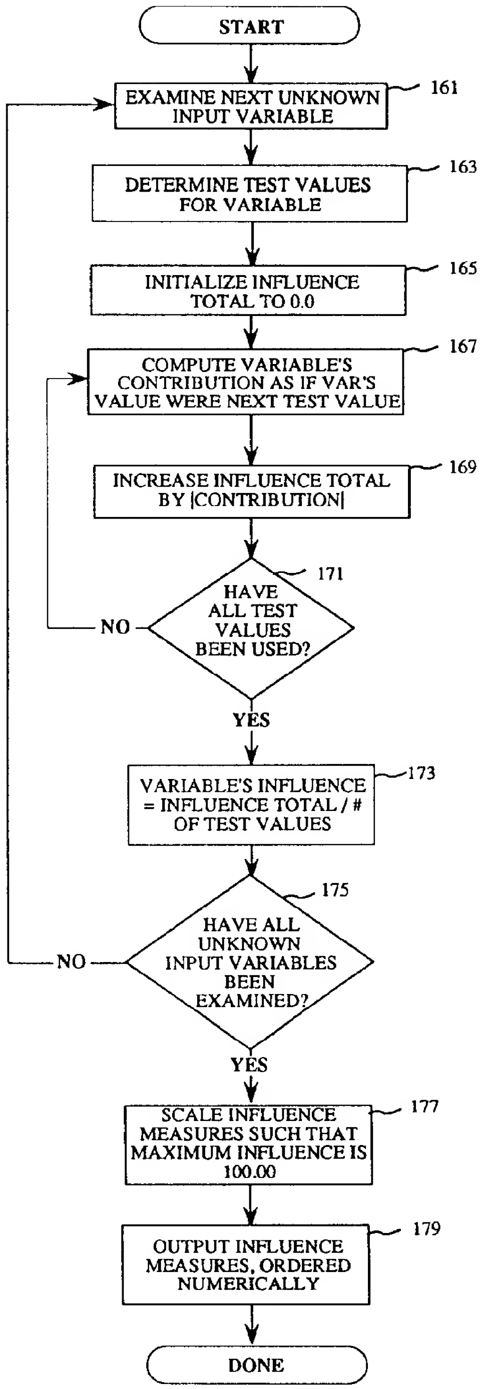
START
EXAMINE NEXT UNKNOWN INPUT VARIABLE
161
DETERMINE TEST VALUES FOR VARIABLE
163
INITIALIZE INFLUENCE TOTAL TO 0.0
165
COMPUTE VARIABLE'S CONTRIBUTION AS IF VAR'S VALUE WERE NEXT TEST VALUE
167
INCREASE INFLUENCE TOTAL BY |CONTRIBUTION|
169
HAVE ALL TEST VALUES BEEN USED?
171
NO
YES
VARIABLE'S INFLUENCE = INFLUENCE TOTAL / # OF TEST VALUES
173
HAVE ALL UNKNOWN INPUT VARIABLES BEEN EXAMINED?
175
NO
YES
SCALE INFLUENCE MEASURES SUCH THAT MAXIMUM INFLUENCE IS 100.00
177
OUTPUT INFLUENCE MEASURES, ORDERED NUMERICALLY
179
DONE

Figure 5

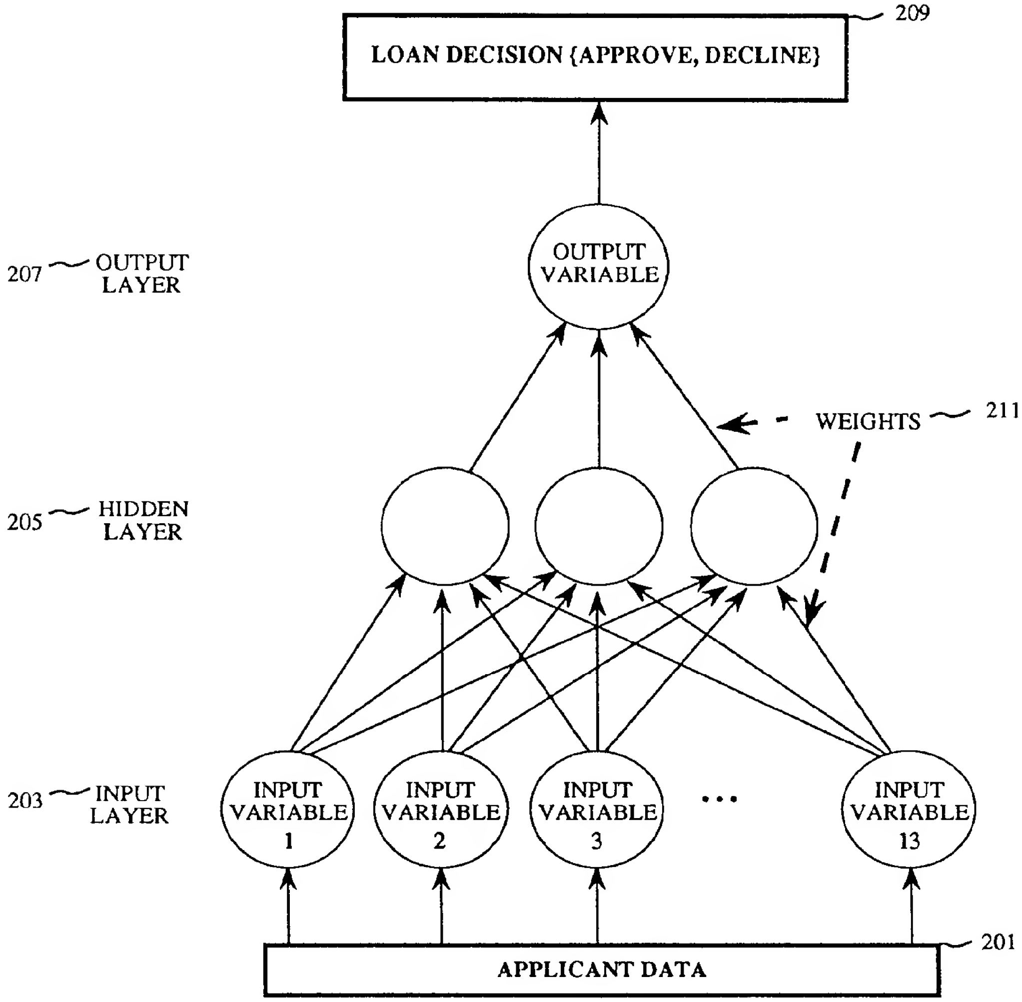
209
LOAN DECISION {APPROVE, DECLINE}
207 OUTPUT LAYER
OUTPUT VARIABLE
WEIGHTS 211
205 HIDDEN LAYER
203 INPUT LAYER
INPUT VARIABLE 1
INPUT VARIABLE 2
INPUT VARIABLE 3
...
INPUT VARIABLE 13
201
APPLICANT DATA

Figure 6

'?' indicates a missing value.
Accounts' refers to sources of credit such as loans and credit cards.

VARIABLE NAME	APPLICANT'S VALUES
Active Accounts (# of accounts currently open)	1
New Accounts (# of accounts opened in the last 18 months)	0
# of Inquiries (# of inquiries made into applicant's credit history in the last 6 months)	2
Public Record Items (# of derogatory items in the credit report, e.g., bankruptcies and tax liens)	0
Bankcard Accounts (# of bank credit cards listed as credit references)	1
Retail Accounts (# of retail store credit cards listed as credit references)	?
30-Days Overdue Now (# of accounts for which payment is currently 30 or more days overdue)	0
90-Days Overdue Ever (# of accounts for which payment has ever been 90 or more days overdue)	0
% Ever 60-Days Overdue (percentage of accounts for which payment has ever been 60 or more days overdue)	0.0
Applicant's Age (age in years)	40
Time at Address (months at current home address)	276
Time at Job (months at present job)	?
Time in Credit File (months in credit-reporting agency's file)	25

Figure 7

VARIABLE	VALUE	CONTRIBUTION TO LOAN APPROVAL
Supporting Factors:		
New Accounts	0	100.0
30-Days Overdue Now	0	49.8
% Ever 60-Days Overdue	0.0	34.9
90-Days Overdue Ever	0	24.0
Insignificant Factors:		
Bankcard Accounts	1	18.8
Time at Address	276	13.6
Public Record Items	0	5.3
Applicant's Age	40	5.1
Active Accounts	1	-1.2
# of Inquiries	2	-6.7
Weakening Factors:		
Time in Credit File	25	-49.7

Figure 8

VARIABLE	VALUE	CONTRIBUTION TO LOAN APPROVAL
Sufficient:		
30-Days Overdue Now	0	100.0
# of Inquiries	2	96.3
Bankcard Accounts	1	78.0
New Accounts	0	65.6
Active Accounts	1	54.2
% Ever 60-Days Overdue	0.0	44.7
Public Record Items	0	33.1
90-Days Overdue Ever	0	25.8
Time at Address	276	20.9
Applicant's Age	40	17.7
Additional:		
Time in Credit File	25	3.3

Figure 9

VARIABLE	INFLUENCE
Bankcard Accounts	100.0
Time at Job	96.5
Retail Accounts	62.9
Applicant's Age	50.6
Time at Address	31.7

Figure 10

IF New Accounts = LOW

AND 30-Days Overdue Now = LOW

AND % Ever 60-Days Overdue = LOW

AND Active Accounts = MEDIUM

AND Bankcard Accounts = HIGH

AND Time at Job = HIGH

CONCLUDE Approve = TRUE

Figure 11